แปลงตัวเลขให้เป็นตัวอักษร

ยืน ภู่วรวรรณ

การใช้งานทางด้านตัวเลขในทางธุรกิจมีความจำเป็นต้องใช้ตัวอักษรในการเขียนกำกับตัวเลขเพื่อความแน่นอนถูกต้องอีกชั้นหนึ่ง เช่นการเขียนตัวเลขในตั๋วสัญญาใช้เงิน เขียนตัวเลขในเอกสารการเงินต่าง ๆ เช่น 10,202,051.75 ก็เขียนเป็นตัวเลขกำกับว่าสิบล้านสองแสนสองพันห้าสิบเอ็ดบาทเจ็ดสิบห้าสตางค์ การเปลี่ยนตัวเลขให้เป็นตัวอักษรนี้จึงเป็นเรื่องที่มีความจำเป็น

เริ่มต้นจากแนวความคิด

แนวความคิดเกิดขึ้นกับผู้เขียนเมื่อตกลงใจว่าจะทำวารสารไมโครคอมพิวเตอร์ฉบับนี้เป็นฉบับ NLP-Natural Language Processing ในขั้นต้นผู้เขียนตั้งใจอยากจะหาวิธีการใช้ดีเบสในการทำการกระจายประโยคภาษาธรรมชาติ แต่เห็นว่าผู้ใช้อาจจะนำไปใช้ประโยชน์ได้ยาก ประจวบกับดีเบสทรีมีจุดอ่อนที่ไม่สามารถทำการแก้ปัญหาแบบ recursion ได้ง่ายเหมือนปาสคาลหรือซี ผู้เขียนจึงพยายามคิดดูว่าจะใช้แนวความคิดทางด้าน NLP ให้เข้ากับเล่มได้อย่างไร ในที่สุดผู้เขียนก็มานึกถึงคำถามของผู้ที่เคยโทรศัพท์มาถามผู้เขียนว่าเคยเขียนรูทีนทำการแตกกระจายเงินให้เป็นภาษาพูดบ้างไหม ผู้เขียนก็เลยตกลงใจว่าจะเขียนโปรแกรมดีเบสในการแตกกระจายเงินจากตัวเลขเป็นตัวอักษร และก็มาค้นหาบทความเก่า ๆ ในจดหมายข่าวของบริษัทแอชตัน–เทตในฉบับเก่า ๆ สมัยเมื่อครั้งดีเบสทูของเครื่อง 8 บิต แต่ก็พบว่าการพิจารณาตัวอักษรตัวเลขของไทยกับของต่างประเทศต่างกันมาก เพราะของไทยมี "เอ็ด" มี "ยี่" ตกลงผู้เขียนจึงพยายามทำให้ดูทั้งสองแบบคือทั้งแบบของภาษาอังกฤษซึ่งเอาเค้าโปรแกรมมาจากบางส่วนของจดหมายข่าวของบริษัทแอชตัน–เทต แต่ฉบับภาษาไทยผู้เขียนเขียนเองทั้งหมด และเพื่อหลีกเลี่ยงกับรหัสภาษาไทยและตัวอักษรไทยผู้เขียนจึงใช้หลักการโรมาไนซ์ตัวอักษร เช่น SIB ED BAHT YEE SIB STANGS เพื่อว่าผู้อ่านจะนำไปเปลี่ยนเป็นภาษาไทยเองได้ง่าย และก็ใช้ได้จริงตามที่ต้องการอีกด้วย หลักการที่ผู้เขียนได้ให้ไว้สามารถขยายตัวเลขให้มากขึ้น ในขณะนี้ผู้เขียนจำกัดตัวเลขไว้สูงสุด 99,999,999.99 บาท หรือ 99 ล้าน หากต้องการได้มากกว่านี้ก็เพิ่มเติมโปรแกรมและแก้ไขอีกเพียงเล็กน้อยเท่านั้น ลองดูว่าจะแก้ไขได้อย่างไร

โครงสร้างการแตกกระจายตัวเงินแบบภาษาอังกฤษ

ระบบภาษาอังกฤษจะอ่านตัวเลขตามหลักพัน เช่น 123,451.12 จะอ่านได้ว่า "one hundred twenty-three thousand, four hundred fifty-one bahts and 12 stangs ดังนั้นในระบบภาษาอังกฤษจะมีการกำหนดตัวเลขที่สัมพันธ์กับตัวอักษรดังนี้

การนับหลักหน่วย – one two three four five six seven eight nine ten

การนับสิบถึงยี่สิบ – eleven twelve thirteen fourteen fifteen sixteen seventeen eighteen ninteen

การนับหลักสิบ – ten twenty thirty forty fifty sixty seventy eighty ninty

การแตกตัวเงินให้เป็นตัวอักษรจึงต้องเอาหลักการทางภาษาที่ใช้กันนี้มาเป็นตัวกำหนดค่าตามตัวเลข ในที่นี้จะกำหนดเป็นตัวแปรอะเรย์สามตัวคือ

t_unit = "one two threefour five six seven eight nine ten "

t_teen = "eleven twelve thirteen fourteen fifteen sixteen seventeeneighteen nineteen"

t_decade = "ten twenty thirty forty fifty sixty seventyeighty ninty "

ลักษณะการกระจายจะเป็นไปตามโปรแกรมที่ให้ไว้โดยการเขียนโปรแกรมเพื่อให้ใช้ง่าย โดยผู้ใช้เพียงแต่ส่งผ่านค่า amt_full ซึ่งเป็นตัวเลขให้กับโปรแกรมนี้แล้วโปรแกรมจะส่งค่าเอาต์พุตออกมาเป็นตัวอักษรอยู่ในตัวแปร amt_word และให้ค่า is_error เพื่อบอกสถานภาพของข้อผิดพลาดหากการแปลงมีข้อผิดพลาด

ในการเขียนโปรแกรมด้วยดีเบส–ทรีพลัส เราอาจตั้งเป็น Procedure และกำหนดการ Pass เป็น PARAMETER ของตัว amt_full, amt_word และ is_error ก็ได้ ซึ่งทำให้ผู้ใช้ไม่ต้องพะวงเรื่องโปรแกรมจะมีการใช้ตัวแปรซ้ำกัน

โปรแกรมที่ให้ไว้ในโมดูลแรกชื่อ NMBRWORD ซึ่งจะรับข้อมูลจากโปรแกรมอื่นคือ amt_full ลองพิจารณาตัวโปรแกรมในรูปที่ 1 ดู

รูปที่ 1 โปรแกรม nmbrword.prg

```
. type nmbrword.prg
* Program :Nmbrword.org
* Date : 2 Feb 1988
* Author :Poovarawan
* Coverts a numberic Baht amount (to B999,999.99) to words
*
* Parameters passed:
* name        type   length     description
* --------    ----   -------    -----------------------
* IN:
*   amt_full   N      3-9        Including 2 decimal places.
* OUT:
*   amt_word   C      23-86      Depending on a mount.
*   is error   L      1          .T. if error in conversion.
*
* Expacts: TALK (OFF)
*    sets: EXACT (OFF)
*
* initialize memory variables...
set talk off
T_unit = "ONE  TWO  THREEFOUR FIVE SIX  SEVENEIGHTNINE TEN  "
T_teen = "ELEVEN   TWELVE   THIRTEEN FOURTEEN FIFTEEN  SIXTEEN  SEVENTEENEIGHTEEN NINETEEN "
T_decade = "TEN    TWENTY THIRTY FORTY  FIFTY  SIXTY  SEVENTYEIGHTY NINETY "
```

รูปที่ 1 (ต่อ)

```
amt_word = " "
is_error =.F.
*
* Convert decimal numbers to a string containing
* the stangs amount...
t_stang_str = str((amt_full-int(amt_full))*100,2)
if t_stang_str = " "
 t_stang_str = "0"+substr(t_stang_str,2,1)
endif
* Change the environment for upcoming branches...
set exact on
*
* Conditionnal branch...
do case
 *
 *Branch for amounts too high or too low...
 case amt_full >999999.99 .or. amt_full<0.00
 is_error=.T.
 release all like t_*
 set exact off
 return
 *
 * Branch for zero bahts...
 case amt_full < 1.00
  amt_word = " NO "
  *
  * Branch for other conditions...
  otherwise
  *
  * Convert baht amount to a character string...
  t_amt_str =  str(int(amt_full),6)
  *
  * Branch for hundred thousands...
  if substr(t_amt_str,1,1) > "0"
   T_hunthous = substr(t_amt_str,1,1)
   amt_word = amt_word + trim(substr(t_unit,(val(t_hunthous)-1)*5+1,5))+"HUNDRED "
  endif
  *
  *Branch for ten-thousands and thousands...
  if substr(t_amt_str,2,2) > " "
  t_tenthous = substr(t_amt_str,2,1)
  t_thousand = substr(t_amt_str,3,1)
  *
  *Branch for combinations of  ten-thosands
  * and thousands...
  do case
  case substr(t_amt_str,1,1) > "0" .and. val(substr(t_amt_str,2,2)) = 0
   amt_word =  amt_word + "THOUSAND"
  case t_tenthous=" " .or. t_tenthous="0"
   amt_word = amt_word+trim(substr(t_unit,(val(t_thousand)-1)*5+1,5))+"THOUSAND"
  case t_thousand = "0"
   amt_word = amt_word+trim(substr(t_decade,(val(t_tenthous)-1)*7+1,7))+"THOUSAND"
```

รูปที่ 1 (ต่อ)

```
 case t_tenthous="1"
  amt_word = amt_word+trim(substr(t_teen,(val(t_thousand)-1)*9+1,9))+ "THOUSAND"
 case t_tenthous>"1"
  a=amt_word+trim(substr(t_decade,(val(t_tenthous)-1)*7+1,7))+"-"+trim(substr(t_unit,(val(t_thousand)-1)*5+1,5))+"THOUSAND"
  amt_word=a
 ENDCASE
 *
 * Branch for comma or space after thousands...
 if val(substr(t_amt_str,4,3))>0
  amt_word = amt_word +","
 else
  amt_word = amt_word +" "
 endif
endif
*
* Branch for hundreds...
if substr(t_amt_str,4,1) > "0"
 t_hundred = substr(t_amt_str,4,1)
 amt_word= amt_word+trim(substr(t_unit,(val(t_hundred)-1)*5+1,5))+ "HUNDRED "
endif
*
* Branch for ten and ones...
if val(substr(t_amt_str,5,2))>0
 t_tens = substr(t_amt_str,5,1)
 t_ones = substr(t_amt_str,6,1)
 *
 * Branch for combinations of tens and ones...
 do case
 case t_tens=" " .or. t_tens="0"
  amt_word= amt_word +trim(substr(t_unit,(val(t_ones)-1)*5+1,5))+ " "
 case t_ones="0"
  amt_word= amt_word +trim(substr(t_decade,(val(t_tens)-1)*7+1,7))+ " "
 case t_tens="1"
  amt_word= amt_word +trim(substr(t_teen,(val(t_ones)-1)*9+1,9))+ " "
 case t_tens>"1"
  amt_word =amt_word +trim(substr(t_decade,(val(t_tens)-1)*7+1,7))+"-"+ trim(substr(t_unit,(val(t_ones)-1)*5+1,5))+ " "
 endcase
endif
endcase
*
* Branch for one baht or more,
* and put the word string together...
if amt_word=" one"
amt_word= substr(amt_word,2,len(trim(amt_word))+1) + "BAHT and "+ t_stang_str + " STANGS"
else
amt_word= substr(amt_word,2,len(trim(amt_word))+1)+"BAHTS and "+ t_stang_str + " STANGS"
endif
*
* Restore theenvironment and return to the calling program...
release all like t:*
set exact off
return
```

โปรแกรมที่เขียนนี้ได้ใส่ comment ไว้มากอาจทำให้การรันโปรแกรมช้าลง เมื่อใช้งานจริงอาจต้องตัด comment ออกจะได้เร็วขึ้น

โปรแกรมเริ่มจากการสร้าง t_stang_str เป็นตัวแปรเก็บเศษสตางค์ และ t_amt_str เป็นตัวแปรเก็บจำนวนตัวเลขที่เป็นจำนวนเต็ม การตรวจสอบตัวเลขถ้าหากว่าตัวเลขมากกว่า 999999.99 หรือน้อยกว่า 0 เราจะถือว่าตัวเลขนี้มีข้อผิดพลาดจะส่ง is_error ออกไป

ในการเรียกใช้โปรแกรมเราจะเขียนโปรแกรมแยก โดยใช้ชื่อโปรแกรม nmbrcall.prg ในรูปที่ 2 โปรแกรมนี้จะเรียก nmbrword แล้วนำค่าที่ได้มาพิมพ์ให้ดูบนจอ

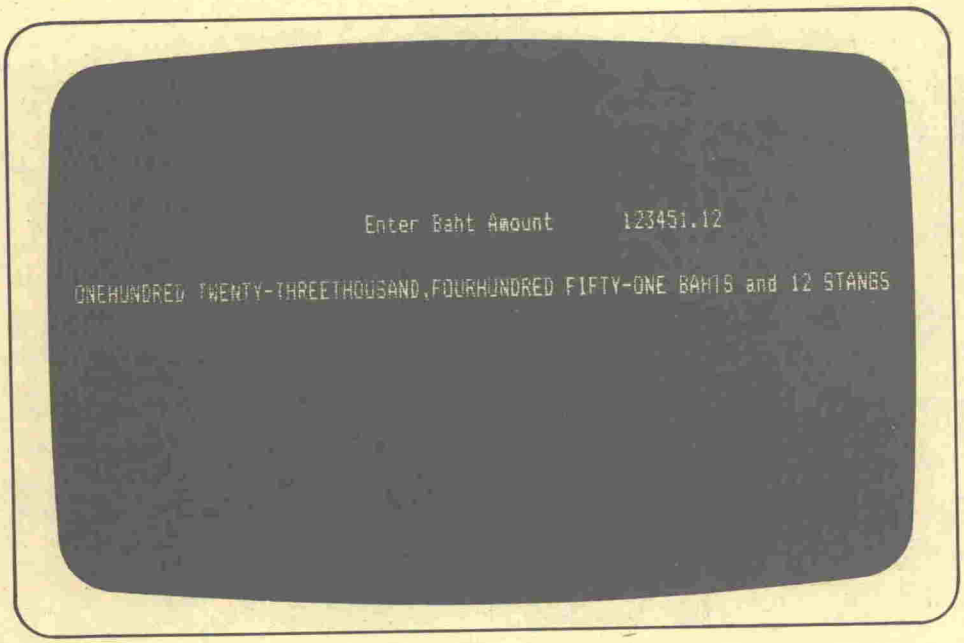
ผลของการรันโปรแกรม nmbrcall และกรอกข้อมูลตัวเลขลงไปจะได้ผลบนจอภาพดังรูปที่ 3

ระบบภาษาไทย

การแตกกระจายตัวเลขระบบภาษาไทยแตกต่างจากภาษาอังกฤษโดยสิ้นเชิง

```
. type nmbrcall.prg
* NUBRCALL.PRG
* Example of code used to call the nmbrword subroutine...
*
* Establish working environment,and promt for an amount...
clear
set talk off
amt_full=0.00
@ 10,26 say "Enter Baht Amount " get amt_full
read
*
* Call the subrubroutine to do the conversion...
amt_word=
is_error=.t.
do nmbrword
*
* Branch for error or not...
if is_error
   @ 12,0 say "ERROR! Number must be 0.00 to 999,999.99"
else
   * Do anything you want to her with the result...
   * Optional display...
   @ 12,0 say amt_word
endif
*
* Restore the environment and close this command file...
release amt_full,amt_word,is_error
set talk on
return
* eof:Nmbrcall.prg
```

รูปที่ 2 โปรแกรม nmbrcall.prg



รูปที่ 3
ผลการรันโปรแกรม nmbrcall

เพราะภาษาไทยมีลักษณะเฉพาะ เช่น 11 อ่านว่า *สิบเอ็ด* 12 อ่านว่า *สิบสอง* 101 อ่านว่า *หนึ่งร้อยเอ็ด* หรือ 21 อ่านว่า *ยี่สิบเอ็ด* ระบบภาษาไทยจึงเกี่ยวพันกับตัวเลข 1 และ 2 คือในหลักหน่วยจะเป็นเอ็ด และ 2 ในหลักสิบจะเป็นยี่แต่การอ่านภาษาไทยจะไม่อ่านเป็นพันเหมือนภาษาอังกฤษแต่จะอ่านเป็นล้านคือภาษาไทย มี *แสน หมื่น พัน ร้อย สิบ* และถ้าเกินล้านก็จะอ่านทบกลับมา เช่น 21,101001.11 ก็อ่านว่า *ยี่สิบเอ็ดล้านหนึ่งแสนหนึ่งพันเอ็ดบาทสิบเอ็ดสตางค์* ระบบภาษาไทยจึงมีลักษณะเฉพาะอยู่ที่หลักหน่วยคือ 1 หลักสิบคือ 2 และหลักล้านคือ 1 หลักสิบล้านคือ 2 เช่นกัน การเขียนโปรแกรมของการแตกกระจายเงินเราก็ทำคล้ายกัน โดยเริ่มแยกหลักบาทและสตางค์ออกจากกัน ถ้าไม่มีเศษสตางค์เราจะใส่คำว่าถ้วนลงไป เช่น 100.00 ก็จะเขียนว่า *หนึ่งร้อยบาทถ้วน*

ตัวแปรหลักที่จะนำมาใช้จึงประกอบด้วย

t_unit = "NUNGSONGSAM SEE HA HOK CHEDPAD KHAO SIB "

การแปลงจะแปลงเศษสตางค์ก่อนโดยแยกเศษสตางค์เป็นข้อความแล้วนำมาตรวจสอบเมื่อใส่ตัวเลข หลังจากนั้นก็จะแปลงจำนวนบาท การแปลงจำนวนบาทเรามีลูปการทำงาน 8 ลูป เพื่อตรวจสอบตัวเลขจากหลักหน่วยไปหาหลักสิบล้านทีละหลักแล้วใส่ข้อความเติมด้านหน้าไปเรื่อย ๆ

โปรแกรมที่เขียนนี้นำเอาหลักการส่งผ่านตัวแปรของโปรแกรมก่อนมาใช้ คือส่งค่ามาด้วยตัวแปร amt_full และส่งกลับด้วย amt_word และ is_eror โปรแกรมที่เขียนขึ้นใช้ชื่อโปรแกรมว่า numpar ในรูปที่ 4

รูปที่ 4 โปรแกรม numpar.prg

```
. type numpar.prg
* Program :Numpar.prg
* Date : 2 Feb 1988
* Author :Poovarawan
* Coverts a numberic Baht amount (to B999,999.99) to words
*
* Parameters passed:
* name         type   length     description
* --------     ----   -------    ------------------------
* IN:
*   amt_full   N      3-12        Including 2 decimal places.
* OUT:
*   amt_word   C      23-86       Depending on a mount.
*   is_error   L      1           .T. if error in conversion.
*
* Expacts: TALK (OFF)
*    sets: EXACT (OFF)
*
* initialize memory variables...
set talk off
T_unit = 'NUNGSONGSAM SEE HA  HOK CHEDPAD KHAOSIB '
ed  = ' ED'
yee  = ' YEE'
amt_word = " "
is_error =.F.
*
* Convert decimal numbers to a string containing
* the stangs amount...
t_stang_str = str((amt_full-int(amt_full))*100,2)
if t_stang_str = " "
 tstang_val=val(substr(t_stang_str,2,1))
 if tstang_val=0
  t_stang_str=' THAUN'
 else
  t_stang_str=substr(T_unit,(tstang_val*4)-3,4)+' STANGS'
 endif
else
 stang_val1=val(substr(t_stang_str,1,1))
 stang_val2=val(substr(t_stang_str,2,1))
 if stang_val1=1
  t_stang_str = ' SIB '
 else
  if stang_val1=2
   t_stang_str = yee+' SIB '
  else
   t_stang_str = substr(T_unit,(stang_val1*4)-3,4) + ' SIB '
  endif
 endif
 if stang_val2>0
  if stang_val2=1
   t_stang_str=t_stang_str+ed + ' STANGS'
  else
   t_stang_str=t_stang_str+substr(T_unit,(stang_val2*4)-3,4) + ' STANGS
```

รูปที่ 4 (ต่อ)

```
  endif
 else
  t_stang_str=t_stang_str+' STANGS'
 endif
endif
amt_word=t_stang_str
*
* Conditionnal branch...
do case
 *
 *Branch for amounts too high or too low...
 case amt_full >99999999.99 .or. amt_full<0.00
 is_error=.T.
 release all like t_*
 set exact off
 return
 *
 * Branch for zero bahts...
 case amt_full < 1.00
  amt_word = 'NO  BAHT '+amt_word
  *
  * Branch for other conditions...
  otherwise
  *
  * Convert baht amount to a character string...
  t_amt_str =  str(int(amt_full),8)
  *
  i=1
  * Parsing  8 digit number by parser loop
  * We can not use recursion on dBASE III+
  * So 8 loops will be used
  do while i<= 8
   numstr=substr(t_amt_str,9-i,1)
   if int(amt_full)=1
    amt_word= substr(T_unit,1,4)+' BAHT '+amt_word
    i=9
   else
    if numstr>'0'
     sub_word= substr(T_unit,val(numstr)*4-3,4)
    else
     sub_word=''
    endif
    do case
    * Units number or lak noui
    case i=1
     if amt_full>=10 .and. numstr='1'
      amt_word= ed+' BAHT ' +amt_word
     else
      amt_word=sub_word+' BAHT '+amt_word
     endif
    * Lak sib
    case i = 2 .and. numstr>'0'
     if numstr='1'
      amt_word=' SIB ' + amt_word
     else
      if numstr='2'
       amt_word=yee+' SIB '+amt_word
      else
       amt_word=sub_word+' SIB '+amt_word
      endif
     endif
    * lak roy
    case i=3  .and. numstr > '0'
     amt_word=sub_word+ ' ROY '+amt_word
    * lak pan
    case i = 4 .and. numstr>'0'
     amt_word=sub_word+' PAN '+amt_word
    * lak muan
    case i=5  .and. numstr > '0'
     amt_word=sub_word+' MUAN '+amt_word
    * lak sann
    case i = 6 .and. numstr>'0'
     amt_word=sub_word+' SAN '+amt_word
    * lak lan
    case i=7  .and. numstr > '0'
     if amt_full>=10000000 .and. numstr='1'
      amt_word= ed+' LAN ' +amt_word
     else
      amt_word=sub_word+' LAN '+amt_word
     endif
    * lak sib lan
    case i = 8 .and. numstr>'0'
     if numstr='1'
      amt_word=' SIB ' + amt_word
     else
      if numstr='2'
       amt_word=yee+' SIB '+amt_word
      else
       amt_word=sub_word+' SIB '+amt_word
      endif
     endif
    endcase
    i=i+1
   endif
  enddo
 endcase
*
* Restore the environment and return to the calling program...
release all like t_*
set exact off
return
*eof: Numpar.prg
```

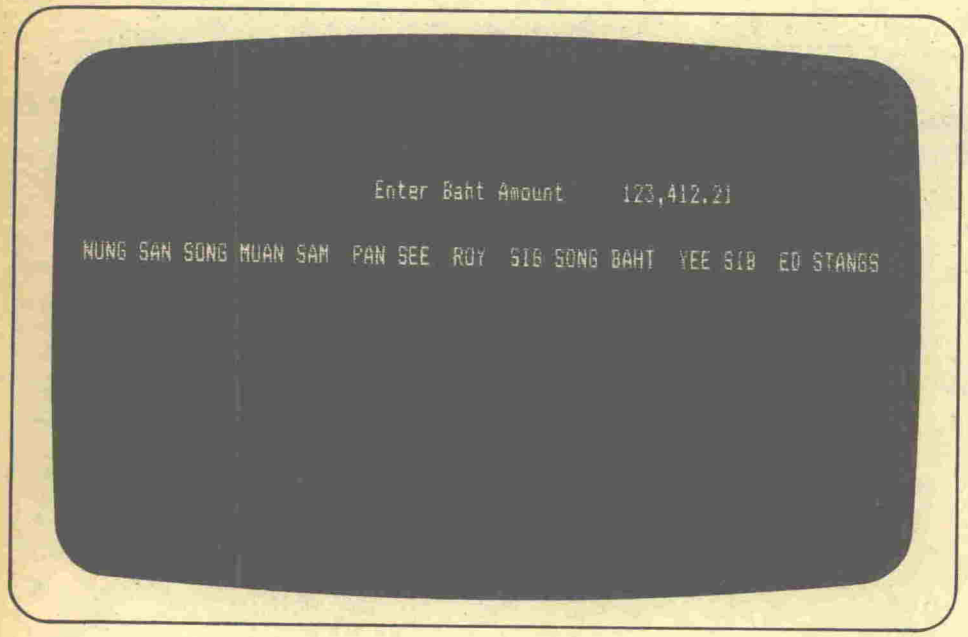
```
.
.
. type numcall.prg
* NUMCALL.PRG
* Example of code used to call the nmbrword subroutine...
*
* Establish working environment,and promt for an amount....
clear
set talk off
amt_full=0.00
@ 10,26 say "Enter Baht Amount " get amt_full pict '99,999,999.99'
read
*
* Call the subrubroutine to do the conversion...
amt_word=' '
is_error=.t.
do numpar
*
* Branch for error or not...
if is_error
   @ 12,0 say "ERROR! Number must be 0.00 to 99,999,999.99"
else
   * Do anything you want to her with the result...
   * Optional display...
   @ 12,0 sav amt_word
endif
*
* Restore the environment and close this command file...
release amt_full,amt_word,is_error
set talk on
return
* eof:Numcall.prg
```

รูปที่ 5 โปรแกรม numcall.prg

รูปที่ 5 เป็นตัวอย่างโปรแกรมเรียกใช้ชื่อ numcall เมื่อรันโปรแกรมและป้อนข้อมูลเข้าไปผลที่เกิดขึ้นจะเป็นดังรูปที่ 6

บทสรุป

รายละเอียดของโปรแกรมผู้เขียนได้ใส่ comment ไว้เพื่อให้อ่านเข้าใจได้ง่าย ผู้ใช้สามารถนำโปรแกรมย่อยนี้ไปใช้ประโยชน์ได้โดยการเรียกและส่งค่ามาให้ อย่างไรก็ตาม ผู้เขียนจำกัดตัวเลขไว้เพียง 99,999,999.99 ซึ่งคิดว่าส่วนใหญ่จะพอเพียงต่อการใช้งาน หากยังไม่พอก็สามารถเพิ่มขยายได้โดยไม่ยากโดยการเพิ่มลูป i และใส่ case เพิ่มอีก ลองดูก็ได้ครับ แล้วพบกันใหม่ในฉบับหน้า



รูปที่ 6
ผลการรันโปรแกรม numcall